A Walk in the Clouds

Routing through VNFs on Bidirected Networks

Förster, Klaus-Tycho; Parham, Mahmoud; Schmid, Stefan

Published in:
Algorithmic Aspects of Cloud Computing

DOI (link to publication from Publisher):
10.1007/978-3-319-74875-7_2

Publication date:
2018

Document Version
Accepted author manuscript, peer reviewed version

The final authenticated publication is available online at https://doi.org/10.1007/978-3-319-74875-7_2

A Walk in the Clouds: Routing through VNFs on Bidirected Networks

Klaus-Tycho Foerster, Mahmoud Parham, and Stefan Schmid

Aalborg University, Denmark
{ktfoerster,mahmoud,schmiste}@cs.aau.dk

Abstract. The virtualization of network functions enables innovative new network services which can be deployed quickly and at low cost on (distributed) cloud computing infrastructure. This paper initiates the algorithmic study of the fundamental underlying problem of how to efficiently route traffic through a given set of Virtualized Network Functions (VNFs). We are given a link-capacitated network $G = (V, E)$, a source-destination pair $(s, t) \in V^2$ and a set of waypoints $\mathscr{W} \subset V$ (the VNFs). In particular, we consider the practically relevant but rarely studied case of bidirected networks. The objective is to find a (shortest) route from s to t such that all waypoints are visited. We show that the problem features interesting connections to classic combinatorial problems, present different algorithms, and derive hardness results.

1 Introduction

After revamping the server business, the virtualization paradigm has reached the shores of communication networks. Computer networks have broken with the "end-to-end principle" [37] a long time ago, and today, intermediate nodes called *middleboxes* serve as proxies, caches, wide-area network optimizers, network address translators, firewalls, etc. The number of middleboxes is estimated to be in the order of the number of routers [21].

The virtualization of such middleboxes is attractive not only for cost reasons, but also for the introduced flexibilities, in terms of fast deployment and innovation: in a modern and virtualized computer network, new functionality can be deployed quickly in virtual machines on cloud computing infrastructure. Moreover, the composition of multiple so-called Virtual Network Functions (VNFs) allows to implement complex network services known as *service chains* [33]: traffic between a source s and a destination t needs to traverse a set of network functions (for performance and/or security purposes).

However, the realization of such service chains poses a fundamental algorithmic problem: In order to minimize the consumed network resources, traffic should be routed along *short* paths, while respecting capacity constraints. The problem is particularly interesting as due to the need to traverse waypoints, the resulting route is not a simple path, but *a walk*.

In this paper, we are particularly interested in VNF routing on *bidirected networks*: while classic graph theoretical problems are typically concerned with

undirected or directed graphs, real computer networks usually rely on links providing full-duplex communication. Figure 1 gives an example.

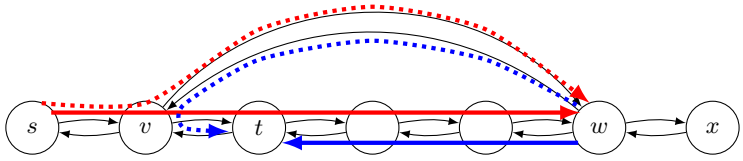

Fig. 1. In this introductory example, the task is to route the flow of traffic from the source s to the destination t via the waypoint w. When routing via the solid red (s, w) path, followed by the solid blue (w, t) path, the combined walk length is $5 + 3 = 8$. A shorter solution exists via the dotted red and blue paths, resulting in a combined walk length of $2 + 2 = 4$. Observe that when the waypoint would be on the node x, no node-disjoint path can route from s to t via the waypoint. Furthermore, some combinations can violate unit capacity constraints, e.g., combining the solid red with the dotted blue path induces a double utilization of the link from v to t.

1.1 Model

We study full-duplex networks, modeled as connected *bidirected graphs* $G = (V, E)$ [13] with $|V| = n$ nodes (switches, middleboxes, routers) and $|E| = m$ links, where each link $e \in E$ has a capacity $c : E \to \mathbb{N}_{>0}$ and a weight $\omega : E \to \mathbb{N}_{>0}$. Bidirected graphs (also known as, e.g., Asynchronous Transfer Mode (ATM) networks [10] or symmetric digraphs [22]) are directed graphs with the property that if a link $e = (u, v)$ exists, there is also an anti-parallel link $e' = (v, u)$ with $c(e) = c(e')$ and $\omega(e) = \omega(e')$.

Given (1) a bidirected graph, (2) a source $s \in V$ and a destination $t \in V$, and (3) a set of k waypoints $\mathscr{W} \subset V$, the *bidirected waypoint routing problem* BWRP asks for a flow-route $\mathscr{R}$ (i.e., a walk) from s to t that (i) visits all waypoints in $\mathscr{W}$ and (ii) respects all link capacities. Without loss of generality, we normalize link capacities to the size of the traffic flow, removing links of insufficient capacity.

BWRP comes in two flavors. In the *unordered* version UBWRP, the waypoints $\mathscr{W}$ can be traversed in any order. In the *ordered* version, OBWRP, the waypoints depend on each other and must be traversed in a pre-determined order: every waypoint w_i may be visited at any time in the walk, and as often as desired (while respecting link capacities), but the route $\mathscr{R}$ must contain a given ordered node sequence $s, w_1, w_2, \ldots, w_k, t$. For example, in a network with stringent dependability requirements, it makes sense first route a packet through a fast firewall before performing a deeper (and more costly) packet inspection.

For both UBWRP and OBWRP, we are interested both in *feasible* solutions (respecting capacity constraints) as well as in *optimal* solutions. In the context of the latter, we aim to optimize the cost $|\mathscr{R}|$ of the route $\mathscr{R}$, i.e., we want to minimize the sum of the weights of all traversed links. As we will see,

computing an optimal route $\mathscr{R}^*$ can be hard in general, and hence, we also study $\alpha-$competitive approximation algorithms, where for any computed tour $\mathscr{R}$ holds: $|\mathscr{R}| \leq \alpha \cdot |\mathscr{R}^*|$.

1.2 Contributions

We initiate the study of the waypoint routing problem on bidirected networks. We put the problem into perspective with respect to classic combinatorial problems (in particular Steiner Tree problems, variants of Traveling Salesman problems, and link-disjoint paths problems), and present a comprehensive set of algorithms and hardness results.

Unordered bidirected waypoint routing UBWRP. We first show that while *any* UBWRP instance is *feasible*, as each link is traversed only once in each direction, computing *optimal* solutions is NP-hard: no polynomial-time approximation scheme (PTAS) exists unless P=NP. On the positive side, by leveraging connections to metric TSP, we show that an ≈ 1.53-approximation is possible on general graphs. But also *optimal* solutions are possible in polynomial time, namely if the number of waypoints is small, namely $k \in \mathcal{O}(\log n)$: a practically very relevant case. In fact, if the network is planar, we can solve the problem even up to $k \in \mathcal{O}(\log^{2-\varepsilon} n)$ many waypoints in polynomial time (for any constant $\varepsilon > 0$), using a connection to Subset TSP.

Ordered bidirected waypoint routing OBWRP. Due to a connection to link-disjoint paths, it holds that feasible routes can be computed in polynomial time for OBWRP if $k \in \mathcal{O}(1)$. Moreover, while finding optimal routes is NP-hard in general, we show that polynomial-time exact solutions exist for cactus graphs with constant link capacities.

1.3 Related Work

While routing through network functions is a standard application in the area of computer networking, we curently witness the emergence of two new trends which change the requirements on routing: (1) (virtualized) network functions are increasingly deployed not only at the edge but also in the network core [14]; (2) network functions are being composed or "chained" to provide more complex services, a.k.a. *service chains* [19],[31],[34]. Traversing these network functions may entail certain detours, i.e., routes do not necessarily follow shortest paths and may even contain loops, i.e., form a *walk* rather than a simple path [5],[15],[16],[30].

Waypoint Routing Amiri et al. [1] recently provided first results on the ordered waypoint routing problem. In contrast to our work, Amiri et al. [1] focus on directed and undirected graphs only and do not consider approximation algorithms; however, finding a feasible solution to the ordered waypoint routing problem is NP-hard on undirected graphs, and for directed graphs already for a single waypoint. The same limitation holds for the work by Amiri et al. [2] on the unordered waypoint problem on undirected graphs. Moreover, under unit

capacities, the undirected problem in [2] also has a different structure, and, e.g., it is not possible to establish the connection to Steiner Tree and Traveling Salesman problem variants as discussed in this paper. On the positive side, it is easy to see that their algorithm to compute optimal unordered solutions on bounded treewidth graphs in XP time also applies to our case; however, their hardness results do not.

Link-Disjoint Paths Closely related to the (ordered) waypoint routing problem is the study of link-disjoint paths: Given k source-destination pairs, is it possible to find k corresponding pairwise link-disjoint paths? For an overview of results on directed and undirected graphs, we refer to the work of Amiri et al. [1].

For bidirected graphs, deciding the feasibility of link-disjoint paths is NP-hard [10]. When the number of link-disjoint paths is bounded by a constant, feasible solutions can be computed in polynomial time [22]. Extensions to multi-commodity flows have been studied in [23],[39] and parallel algorithms were presented in [26],[27]. To the best of our knowledge, the joint optimization (i.e., shortest total length) of a constant number of link-disjoint paths is still an open problem. In comparison, we can solve UBWRP for a super-constant number of waypoints optimally.

Besides simple graph classes such as trees, we are not aware of algorithms for a super-constant number of link-disjoint paths on directed graphs, which are also applicable to bidirected graphs. We refer to [32] for an annotated tableau. In contrast, we in this paper optimally solve OBWRP on cactus graphs with constant capacity.

Traveling Salesman and Steiner Tree problems The unordered problem version is related to the *Traveling Salesman problem* (TSP): there, the task is to find a cycle through all nodes of an undirected graph, which does not permit any constant approximation ratio [12], unless P=NP: for the so-called metric version, where nodes may be visited more than once (identical to the TSP with triangle inequality), performing a DFS-tour on a minimum spanning tree (MST) gives a 2-approximation, see, e.g., [12] again. The best known approximation ratio for the metric TSP is 1.5 [11] ($3/2+1/34 \approx 1.53$ for $s \neq t$, called the $s-t$ *Path TSP* [38]) and no better polynomial-time approximation ratio than $123/122 \approx 1.008$ is possible [24], unless P=NP. Throughout this paper, when referring to (any variant of) TSP, we usually refer to the metric version on undirected graphs.

A related problem is the NP-hard [18] *Steiner Tree problem* (ST) on undirected graphs: given a set of terminals, construct a minimum weight tree that contains all terminals. If all nodes are terminals, this reduces to the (polynomial) MST problem. The currently best known approximation ratios for ST are $\ln 4 + \varepsilon < 1.39$ [9] (randomized) and $1 + \frac{\ln 3}{2} \approx 1.55$ [36] (deterministic).

Note that both the Steiner Tree and the Traveling Salesman problem are *oblivious* to link capacities. Notwithstanding, we can make use of approximation algorithms for both problems for UBWRP in later sections.

Prize-Collecting and Subset variants In particular, if $\mathscr{W} \subsetneq V$, we can utilize the *prize-collecting* versions of (path) TSP and ST, called (PC-PATH2) PCTSP and PCST [3]. Here, every node is assigned a non-negative prize (penalty), s.t. if

the node is not included in the tour/tree, its prize is added to the cost. For all three prize-collecting variants above, Archer et al. [3] provide approximation ratios smaller than 2: (1) PCTSP: $97/49 < 1.979592$, (2) PC-PATH2: $241/121 < 1.991736$, (3) PCST: 1.9672 (randomized) and 1.9839 (deterministic).

Contained in PC-TSP is *Subset TSP*, first proposed in [4, Sec. 6], which asks for a shortest tour through specified nodes [28]. Klein and Marx [29] give a $\left(2^{\mathcal{O}(\sqrt{k}\log k)} + W\right) \cdot n^{\mathcal{O}(1)}$ for planar graphs, where W is the largest link weight. They also point out that classic dynamic programming techniques [6],[20] have a runtime of $2^k \cdot n^{\mathcal{O}(1)}$ for general graphs. The path version is not considered, we show how to apply any optimal Subset TSP algorithm to UBWRP with $s \neq t$.

Conceptually related (to the non-metric TSP) is the K-cycle problem, which asks for a shortest cycle through K specified nodes or links, i.e., a vertex-disjoint tour. Björklund et al. [7] give a randomized algorithm with a runtime of $2^K \cdot n^{\mathcal{O}(1)}$.

1.4 Paper Organization

We begin with studying UBWRP in Sec. 2: after giving an introduction to waypoint routing in Sec. 2.1, we give hardness and (approximation) algorithm results via metric and subset TSP in Sec. 2.2. We then consider the ordered case in Sec. 3, tackling constant k in Sec. 3.1, we study hardness in Sec. 3.2, and finally provide a cactus graph algorithm in Sec. 3.3. Lastly, we conclude in Sec. 4 with a short summary and outlook.

2 The Unordered BWRP

We start our study of bidirected waypoint routing with a short *tour d' horizon* of the problem in Sec. 2.1, discussing the case of few waypoints and first general approximations via the Steiner Tree problem. We then follow-up in Sec. 2.2 by showing that UBWRP and metric TSP are polynomially equivalent, i.e., algorithmic and hardness results can be reduced. We also establish connections to the subset and prize-collecting TSP variants.

2.1 An Introduction to (Unordered) Waypoint Routing

First, we examine the case of a *single waypoint* w, which requires finding a shortest $s-t$ route through this waypoint. Note that for a single waypoint, the two problem variants UBWRP and OBWRP are equivalent.

One waypoint: greedy is optimal. We observe that the case of a single waypoint is easy: simply taking two *shortest paths* (*SP*s) $P_1 = SP(s, w)$ and $P_2 = SP(w, t)$ in a greedy fashion is sufficient, i.e., the route $\mathscr{R} = P_1P_2$ is always feasible (and thus, also always optimal in regards to total weight).

Suppose this is not the case, that is, $P_1 \cap P_2 \neq \emptyset$. Among all nodes in $P_1 \cap P_2$, let u and v be, resp., the first and the last nodes w.r.t. to the order of visits in $\mathscr{R}$. Let P_i^{xy} denote the sub-path connecting x to y in P_i. Thereby we have

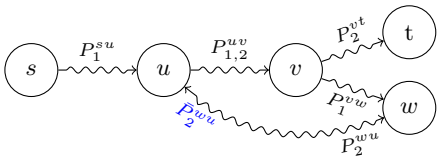

Fig. 2. The directed path from u to v is traversed two times in $\mathscr{R}$.

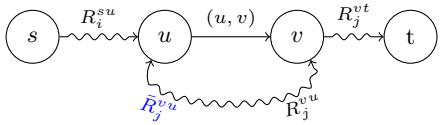

Fig. 3. The link (u, v) is traversed multiple times in $\mathscr{R}$.

$\mathscr{R} = P_1 P_2 = P_1^{su} P_1^{uv} P_1^{vw} P_2^{wu} P_2^{uv} P_2^{vt}$ (Fig. 2). Let $\bar{\mathcal{P}}$ be the reverse of any walk $\mathcal{P}$ obtained by replacing each link $(x, y) \in \mathcal{P}$ with its anti-parallel link (y, x). Observe that for $P_1' = P_1^{su} \bar{P}_2^{wu}$ and $P_2' = \bar{P}_1^{vw} P_2^{vt}$, the feasible route $\mathscr{R}' = P_1' P_2'$ is shorter than $\mathscr{R}$, contradicting P_1 and P_2 both being shortest paths.

Being greedy on the right order is optimal. Next, we show that even more waypoints can be handled efficiently as long as their number is limited: if the optimal traversal order is known, selecting shortest paths in a greedy fashion is again optimal, but to find the optimal order, $k!$ has to be tractable.

We first give an auxiliary lemma, which will also be of later use. We note that Klein and Marx gave an analogous construction for undirected graphs in [29, Figure 1]. The idea can be explained with Fig. 3: when a link (u, v) is traversed at least twice, we can take a shortcut from u to v. Iterating this idea ensures at most one traversal per link.

Lemma 1. *Any (unfeasible) route $\mathscr{R}$ that traverses some link (u, v) more than once, can be efficiently transformed into a shorter route $\hat{\mathscr{R}}$ that traverses each link at most once.*

Proof. There is at least one link that is being shared by at least two sub-routes along $\mathscr{R}$. For every $k, l : k < l$, denote the set of all links $e \in R_k \cap R_l$ by W. Let $(u, v) \in W$ be a link traversed by R_i and R_j s.t. $i < j = \arg\min_l((u, v) \in R_i \cap R_l)$ (Fig. 3). Thus, $\mathscr{R} = R_i R_j = R_i^{su}(uv) R_j^{vu}(uv) R_j^{vt}$. Now consider the new route $\mathscr{R}' = R_1^{su} \bar{R}_2^{vu} R_2^{vt}$. Note that $\mathscr{R}'$ might still traverse (u, v) more than once. Repeat the same procedure for (u, v) until this is not the case. Each time we choose the two sub-routes that preceed any other sub-route traversing (u, v). This ensures the other parts of $\mathscr{R}$ will not be skipped after this transformation. Now we remove this link from W. Since we included additional links for the rerouting, there might be some link $(u', v') \in \bar{R}^{vu} \cap \mathscr{R}, (u', v') \neq (u, v)$, that is being traversed more than once in $\mathscr{R}'$, but not in $\mathscr{R}$. Add all such links to W. Repeating the same procedure for every $(x, y) \in W$ transforms $\mathscr{R}'$ into a new route $\mathscr{R}''$ that traverses (x, y) at most once. On the other hand, for each newly included link we exclude its anti-parallel link during the transformation. Therefore the new route is shorter by at least one link, i.e. $|\mathscr{R}''| < |\mathscr{R}|$. Hence, there can be only $O(|\mathscr{R}|)$ iterations, the last of which necessarily ends up at the desired route $\hat{\mathscr{R}}$ satisfying the lemma. □

Using this idea, we can now show that in an optimal ordering, the shortest paths will not overlap.

Lemma 2. *Given the permutation σ of waypoints in the (first visit) order of an optimal route $\mathscr{R}^*$, we can efficiently construct a route $\mathscr{R}_\sigma$ s.t. $|\mathscr{R}_\sigma| = |\mathscr{R}^*|$.*

Proof. Let $\mathcal{W}_\sigma = w_{\sigma_1} w_{\sigma_2} ... w_{\sigma_k}$ be the order we intend to visit the waypoints, and $\mathscr{R}'$ be an empty route. For each consecutive pair w_{σ_i} and $w_{\sigma_{i+1}}$ add the shortest path links of $SP(w_{\sigma_i}, w_{\sigma_{i+1}})$ and $SP(w_{\sigma_{i+1}}, w_{\sigma_i})$ to $\mathscr{R}'$. We claim that $\mathscr{R}'$ is a feasible, and constitutes an optimal solution to UBWRP. For the sake of contradiction assume this is not the case. Using Lemma 1 we can construct a feasible route $\hat{\mathscr{R}}$ s.t. $|\hat{\mathscr{R}}| < |\mathscr{R}'|$. This implies that for some shortest path $SP(w_{\sigma_i}, w_{\sigma_j}), |i-j| = 1$, when replaced by the path $\hat{\mathscr{R}}^{w_{\sigma_i} w_{\sigma_j}}$, yields a shorter route, contradicting the optimality of $SP(.)$. □

This directly implies the following Theorem 3, which is essentially a brute-force approach. We note that implications from the connections between UBWRP and metric TSP in the next section will improve Theorem 3 in such a way that $k \in \mathcal{O}(\log n)$ becomes tractable.

If the order of visits in $\mathscr{R}^*$ was part of the input, then by Lemma 2 the union of shortest paths between consecutive waypoints would be necessarily a feasible and therefore an optimal solution. Since this is not the case (i.e. we cannot know the optimal order in advance), one can iterate over all permutations of waypoints and apply Lemma 2. Then the best of all these iterations will give an optimal route. Thus, for constants c and $c' = c + c \log c$, there are $k! = \left(\frac{c \log n}{\log \log n}\right)! < (c \log n)^{\left(\frac{c \log n}{\log \log n}\right)} = ((c \log n)^{\log n})^{\frac{c}{\log \log n}} < (n^{c \log c} n^c) \in \mathcal{O}(n^{c'})$ iterations. After multiplying by the cost of $O(k)$ calls to $SP(.)$, the polynomial time follows immediately. Thus we have the following theorem:

Theorem 3. *For $k \in \mathcal{O}\left(\frac{\log n}{\log \log n}\right)$,* UBWRP *is polynomially solvable.*

We now turn our attention to the general case of $k \in \mathcal{O}(n)$. We first establish a connection to (Steiner) tree problems as well as to the Traveling Salesman problem. Subsequently, we will derive stronger results leveraging the metric TSP (Sec. 2.2).

UBWRP is always feasible. Interestingly, UBWRP is always feasible. We begin with the case of $s = t$: first, compute a minimum spanning tree T_U in the undirected version of G, in G_U. Then, traverse T with a DFS-tour starting at s, using every directed link in the bidirected version of T_U exactly once. As every node $v \in V$ will be visited, so will all waypoints $\mathscr{W}$.

The case of $s \neq t$ is similar: Removing the links from the unique $s-t$-path $s, v_1, v_2, \ldots, v_p, t$ decomposes T_U into a forest, still containing all nodes in V. We now traverse as follows: Take a DFS-tour of the tree attached to s, then move to v_1, take a DFS-tour of the tree attached to v_2, ..., until lastly arriving at t, then taking a DFS-tour of the tree attached to t, finishing at t.

Approximations via the Steiner Tree problem. For $\mathscr{W} = V$, the above MST approach directly yields a 2-approximation, cf. TSP in [12]. The 2-approximation fails though when $\mathscr{W} \neq V$: e.g., if only one node is a waypoint, visiting all other nodes can add arbitrarily high costs. However, there is a direct duality to the

Steiner Tree problem: When setting all waypoints (including s, t) as terminals, an optimal Steiner Tree for these terminals in G_U is a lower bound for an optimal solution to UBWRP: taking the link-set of any route $\mathscr{R}$ in G_U contains the links of a Steiner Tree as a subset. Hence, the construction is analogous to the MST one for $\mathscr{W} = V$. As the best known approximation ratio for the Steiner Tree problem are $\ln 4 + \varepsilon < 1.39$ [9] (randomized) and $1 + \frac{\ln 3}{2} \approx 1.55$ [36] (deterministic), we obtain approximation ratios of $2 \ln 4 + \varepsilon < 2.78$ (rand.) and $2 + \ln 3 \approx 3.09$ (det.), for any constant $\varepsilon > 0$.

2.2 Hardness and Improved Approximation

Next, we show that metric TSP (denoted ΔTSP for the remainder of this section for clarity) is equivalent to our problem of UBWRP on general graphs, in the sense that their corresponding optimal solutions have identical cost.

Theorem 4. *Let I be an instance of* UBWRP *on a bidirected graph G and let I' be an instance of the (path) ΔTSP on the metric closure of the corresponding G_U restricted to $\mathscr{W} \cup \{s, t\}$. The cost of optimal solutions for I and I' are identical.*

Proof. We start with $s = t$ for UBWRP. By setting $V = \mathscr{W} \cup \{s, t\}$, the first reduction follows directly. For the other direction, we first construct an instance of ΔTSP. Then, an optimal solution to ΔTSP must imply an optimal solution to UBWRP and vice versa. Let G'_U be the metric closure of G_U restricted to nodes in $\mathscr{W} \cup \{s, t\}$. An optimal TSP cycle C^* in G'_U (after replacing back the shortest path links) corresponds to a route $\mathscr{R}_{C^*}$ on G_U s.t. $|\mathscr{R}_{C^*}| = |C^*|$. Furthermore, $\mathscr{R}_{C^*}$ possibly violates some link capacities in G. Using Lemma 1 we turn $\mathscr{R}_{C^*}$ to a route $\mathscr{R}'_{C^*}$ feasible for UBWRP. We claim that $\mathscr{R}'_{C^*}$ is optimal. Assume this is not the case, i.e. $|\mathscr{R}'_{C^*}| > |\mathscr{R}^*|$. Let σ be the permutation corresponding to the order of waypoints in $\mathscr{R}^*$. By Lemma 2, we can construct a feasible route $\mathscr{R}_\sigma$, such that $|\mathscr{R}_\sigma| = |\mathscr{R}^*|$ and it uses only the shortest path links chosen by $SP(w_{\sigma_i}, w_{\sigma_{(i+1) \mod k}}), 0 \leq i \leq k$. That is, for the cycle C_σ induced by σ on the links of G', we have $|C_\sigma| = \sum_{i=0}^{k} |SP(w_{\sigma_i}, w_{\sigma_{(i+1) \mod k}})| = |\mathscr{R}_\sigma| \overset{\text{Lemma 2}}{=} |\mathscr{R}^*| < |\mathscr{R}'_{C^*}| \leq |\mathscr{R}_{C^*}| = |C^*|$ (by Lemma 1), which contradicts C^* being optimal.

It remains to show, given $\mathscr{R}^*$ and the order of waypoints therein, σ, the cycle C_σ is optimal for ΔTSP (i.e. $|C_\sigma| = |C^*|$). Assume $|C_\sigma| > |C^*|$. As it was shown previously, we can construct a route $\mathscr{R}'_{C^*}$ s.t. $|\mathscr{R}'_{C^*}| \overset{\text{Lemma 1}}{=} |C^*| < |C_\sigma| \overset{\text{Lemma 2}}{=} |\mathscr{R}^*|$, which contradicts the optimality of $\mathscr{R}^*$. The proof construction for $s \neq t$ is analogous, replacing the TSP cycle with a path from s to t. □

No PTAS for UBWRP, but good approximation ratios. As already seen in the proof of Theorem 4, solutions between the corresponding instances of UBWRP and (path) ΔTSP can be efficiently transformed to one of less or identical cost. As such, we can make use of known algorithms and complexity results, resulting in the following two corollaries regarding hardness and approximability:

Corollary 5. UBWRP *is an NP-hard problem, no better polynomial-time approximation ratio than* $123/122 \approx 1.008$ *is possible [24], unless P=NP.*

Corollary 6. *For* $s = t$, UBWRP *can be approximated in polynomial time with a ratio of 1.5 [11]. For* $s \neq t$, *a ratio of* $3/2 + 1/34 \approx 1.53$ *can be obtained [38].*

Relations to $(0, \infty)$-PC-TSP & Subset TSP. In the prize-collecting (PC) variant, the classical Steiner Tree problem can be formulated as a $(0, \infty)$-PC-ST: the terminals must be included (∞), while all other nodes are not relevant (0). In an analogous fashion, one can solve the $(0, \infty)$-PC-(path)-TSP on undirected graphs G_U, where the nodes with ∞ are the waypoints (and s, t).

As such, we can now apply all algorithmic and hardness results from the $(0, \infty)$ variant of the prize-collecting (path) TSP. However, the known results on the prize-collecting version of TSP are weaker than the ones of ΔTSP: this fact is not surprising, as PC-TSP is a generalization of its $(0, \infty)$ variant and ΔTSP.

The $(0, \infty)$-PC-TSP with $s = t$ may also be formulated as the Subset TSP, which asks for a (shortest) closed tour that visits a subset of nodes.

At this point one may wonder why to bother with the Subset TSP, given the parallels between UBWRP and the general metric TSP? As pointed out by Klein and Marx [29], the metric closure can destroy graph properties, e.g., planarity. For a more concrete example, consider the metric closure of a tree with unit link weights, removing unit weight properties. Hence, focusing on Subset TSP allows for algorithms with better approximation ratios on special graph classes.

Leveraging Subset TSP results for UBWRP. Klein and Marx [29] consider the Subset TSP problem as a cycle rather than a path. For planar graphs with k "waypoints", they give an algorithm with a runtime of $\left(2^{\mathcal{O}(\sqrt{k}\log k)} + W\right) \cdot n^{\mathcal{O}(1)}$, where W is the size of the largest link weight. Furthermore, they point out that the classic TSP dynamic programming techniques [6],[20] can be applied to Subset TSP (with $s = t$), solving it optimally in a runtime of $2^k \cdot n^{\mathcal{O}(1)}$.

We show an extension to $s \neq t$, which enables us to use "cycle" algorithms as a black box: create a new node st, which serves as start and endpoint of the cycle, connecting st to two new waypoints w_s, w_t, and in turn w_s to s and w_t to t, where all new links have some arbitrarily large weight γ, see Fig. 4. W.l.o.g., we can assume that an optimal cycle solution of this modified graph starts with the path st, w_s, s. It is left to show that the subsequent tour ends with t, w_t, st: if not, the two nodes w_s, w_t would be traversed three times, which is a contradiction to optimality due to the choice of their incident link weight γ. Observe that instead of setting γ to "∞", $\gamma = W \cdot n^2$ suffices. Hence, we can use the algorithms froms [6],[20], [29] for the path version of subset TSP, and therefore, for UBWRP.

Corollary 7. UBWRP *can be solved optimally in a runtime of* $2^k \cdot n^{\mathcal{O}(1)}$ *for general graphs and in a runtime of* $\left(2^{\mathcal{O}(\sqrt{k}\log k)} + W\right) \cdot n^{\mathcal{O}(1)}$, *where* W *is the maximal link weight, for planar graphs.*

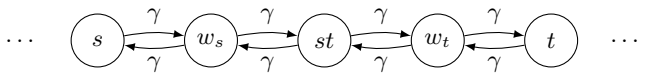

Fig. 4. Illustration of how to add the node st to the graph, connecting it to both s and t via w_s and w_t, respectively. Observe that a tour must traverse both w_s, w_t.

I.e., on general graphs, setting $k \in \mathcal{O}(\log n)$ is polynomial. For planar graphs, we can fix any $0 < x < 1$ with $k \in \mathcal{O}(\log^{1+x} n)$, $W \in n^{\mathcal{O}(1)}$ for polynomiality.

3 Ordered BWRP

The ordered version of BWRP turns out to be quite different in nature from the unordered one. First, we observe that while every BWRP instance is feasible, there are infeasible OBWRP instances due to capacity constaints. E.g., consider Figure 5 with unit capacities and two waypoints. A special case is $k = 1$, which is identical to the unordered case, i.e., routing via one waypoint is always feasible and can be solved optimally in polynomial time.

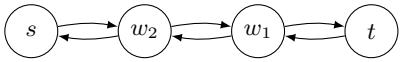

Fig. 5. In this unit capacity network, the task is to route the flow of traffic from s to w_1, then to w_2, and lastly to t. To this end, the link from w_2 to w_1 must be used twice.

In the following, we study OBWRP in three contexts: (1) polynomial-time algorithms for computing feasible routes if $k \in \mathcal{O}(1)$ (Sec. 3.1), NP-hardness of optimality (Sec. 3.2), and optimality on cactus graphs, for any number of waypoints and constant capacities (Sec. 3.3). The latter is practically motivated by the fact that real computer networks often have specific topologies, and especially router-level wide-area topologies are usually quite sparse.

3.1 A Constant Number of Waypoints is Feasible

There is a direct algorithmic connection from the link-disjoint path problem to OBWRP with unit capacities. By setting $s_1 = s$, $t_1 = w_1$, $s_2 = w_1$, $t_2 = w_2$, ..., a $k + 1$ link-disjoint path algorithm also solves unit capacity OBWRP for k waypoints. This method can be extended to general capacities via a standard technique, by replacing each link of capacity $c(e)$ with $\lfloor c(e) \rfloor$ parallel links of unit capacity and identical weight. To get rid of the parallel links, replace each link with a path of length two by "placing" a node on it, with the path weight being identical to the link weight.

Hence, we can apply the algorithm from Jarry and Prennes [22], which solves the feasibility of the link-disjoint path problem on bidirected graphs for a constant number of paths in polynomial runtime.

Theorem 8. *Let $k \in \mathcal{O}(1)$. Feasible solutions for* OBWRP *can be computed in polynomial time.*

The optimal solution already for few link-disjoint paths still puzzles researchers on bidirected graphs, but the problem seems to be non-trivial on undirected graphs as well: while feasibility for a constant number of link-disjoint paths is polynomial in the undirected case as well [25],[35], optimal algorithms for 3 or more link-disjoint paths are not known, and even for 2 paths the best result is a recent randomized high-order polynomial time algorithm [8]. For directed graphs, already 2 link-disjoint paths pose an NP-hard problem [17]. Results for waypoint routing on directed and undirected graphs are analogous [1].

3.2 Optimally Solving OBWRP is NP-Hard

If we transition from a constant number to an arbitrary number of waypoints, we can show that then solving OBWRP optimally becomes NP-hard:

Theorem 9. *Solving* OBWRP *optimally is NP-hard.*

Proof. Reduction from the NP-hard link-disjoint path problem on bidirected graphs $G = (V, E)$ [10]: given k source-destination node-pairs (s_i, t_i), $1 \leq i \leq k$, are there k corresponding pairwise link-disjoint paths?

For every such instance I, we create an instance I' of OBWRP as follows, with all unit capacities: Set $s = s_1$ and $t = t_k$, also setting waypoints as follows: $w_1 = t_1$, $w_3 = s_2$, $w_4 = t_2$, $w_6 = s_3$, $w_7 = t_3$, $\ldots$, $w_{3k-3} = s_k$. We also create the missing $k-1$ waypoints w_2, w_5, w_8, $\ldots$, w_{3k-4} as new nodes and connect them as follows, each time with bidirected links of weight γ: w_2 to $w_1 = t_1$ and $w_3 = s_2$, w_5 to $w_4 = t_2$ and $w_6 = s_3$, $\ldots$, w_{3k-4} to $w_{3k-3} = s_k$ and $w_{3k-5} = t_{k-1}$. I.e., we sequentially connect the end- and start-points of the paths.

Observe that OBWRP is feasible on I' if I is feasible: We take the k link-disjoint paths from I and connect them via the $k-1$ new nodes in I'.

We now set γ to some arbitrarily high weight, e.g., $3k$ times the sum of all link weights. I.e., it is cheaper to traverse every link of I even $3k$ times rather than paying γ once. As thus, if I is feasible, the optimal solution of I' has a cost of less than $2 \cdot k \cdot \gamma$.

Assume I is not feasible, but that I' has a feasible solution $\mathscr{R}$. Observe that a feasible solution of I' needs to traverse the $k-1$ new waypoints, i.e., has at least a cost of $2(k-1)\gamma$. As I was not feasible, we will now show that traversing every new waypoint $w_2, w_5, \ldots$ only once is not sufficient for a feasible solution of I'. Assume for contradiction that one traversal of $w_2, w_5, \ldots$ suffices: for each of those traversals of such a w_j, it holds that it must take place after traversing all waypoints with index smaller than j. Hence, we can show by induction that the removal of the links incident to the waypoints $w_2, w_5, \ldots$ from $\mathscr{R}$ contains a feasible solution for I. As thus, at least one of the waypoints $w_2, w_5, \ldots$ must be traversed twice, i.e., $\mathscr{R}$ has a cost of at least $2 \cdot k \cdot \gamma$.

We can now complete the polynomial reduction, by studying the cost (feasibility) of an optimal solution of I': if the cost is less than $2 \cdot k \cdot \gamma$, I is feasible, but if the cost is at least $2 \cdot k \cdot \gamma$ (or infeasible), I is not feasible. □

While many OBWRP instances are not feasible (already in Figure 5), we conjecture that the feasibility of OBWRP with arbitrarily many waypoints is NP-hard as well. This conjecture is supported by the fact that the analogous link-disjoint feasibility problems are NP-hard on undirected [18], directed [17], and bidirected graphs [10], also for undirected and directed ordered waypoint routing [1]. We thus turn our attention to special graph classes.

3.3 Optimality on the Cactus with Constant Capacity

The difficulty of OBWRP lies in the fact that the routing from w_i to w_{i+1} can be done along multiple paths, each of which could congest other waypoint connections. Hence, it is easy to solve OBWRP optimally (or check for infeasibility) on trees, as each path connecting two successive waypoints is unique.

Lemma 10. OBWRP *can be solved optimally in polynomial time on trees.*

For multiple path options, the problem turns NP-hard though (Theorem 9). To understand the impact of already two options, we follow-up by studying rings.

Lemma 11. OBWRP *is optimally solvable in polynomial time on bidirected ring graphs where for at least one link e holds: $c(e) \in \mathcal{O}(1)$.*

Proof. We begin our proof with $c(e) = c(e') = 1$. Observe that every routing between two successive waypoints has two path options P, clockwise or counterclockwise. We assign one arbitrary path P_e to traverse e, and another arbitrary path $P_{e'}$ to traverse e'. By removing the fully utilized e and e', the remaining graph is a tree with two leaves, where all routing is fixed, cf. Lemma 10.

We now count the path assignment possibilities for e, e': by also counting the "empty assignment", we have at most $(n+1)n$ options, where the optimal routing immediately follows for each option. For these $\mathcal{O}(n^2)$ possibilities, we pick the shortest feasible one. I.e., OBWRP can be solved optimally in polynomial time on rings with unit capacity. To extend the proof to constant capacities $c(e) \in \mathcal{O}(1)$, we use an analogous argument, the number of options for assignments to e and e' are now $\mathcal{O}\left(n^{2c(e)}\right) \in \mathrm{P}$. As thus, the lemma statement holds. □

We now focus on the important case of cactus networks. Our empirical study using the Internet Topology Zoo[1] data set shows that over 30% are *cactus graphs.*

Theorem 12. OBWRP *is optimally solvable in polynomial time on cactus graphs with constant capacity.*

Proof. The idea is to 1) shrink the cactus graph down to a tree, 2) see if for the relevant subset of waypoints (to be described shortly) the feasibility holds on that tree, 3) reincorporate the excluded rings and find the optimal choice of path segments within each ring, and 4) construct an optimal route by stitching together the sub-routes obtained from the tree and the segments from each ring.

[1] See http://www.topology-zoo.org/.

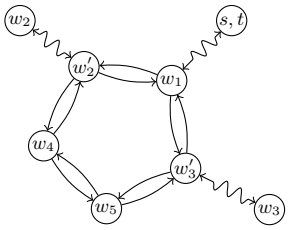

Fig. 6. In this cactus graph, we illustrate the algorithm of Theorem 12 w.r.t. the permutation $w_1w_2w_3w_4w_5$.

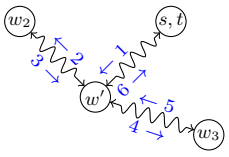

Fig. 7. Once the ring links are contracted, w' replaces the whole ring. Consequently, the permutation reduces to $w'w_2w_3$. The sub-routes are numbered sequentially.

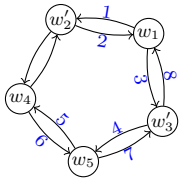

Fig. 8. The permutation induced on the ring is $w_1w'_2w'_3w_4w_5$. In the subproblem, we have $s = t = w_1$. The numbers represent the order of node traversal in the optimal route.

Let $\mathcal{C}$ be the cactus graph (Fig. 6) and $T_{\mathcal{C}}$ be the tree obtained after contracting all the links on each rings. As a result of this link contraction, those waypoints previously residing on rings are now replaced by new (super) waypoints in $T_{\mathcal{C}}$ (Fig. 7). Each super node represents either a subtree of adjacent rings or just an isolated ring. Let $\mathscr{W}'$ denote the waypoints in $T_{\mathcal{C}}$.
Observe that any feasible route in $\mathcal{C}$ through $\mathscr{W}$ corresponds to one unique feasible route in $T_{\mathcal{C}}$ through nodes in $\mathscr{W}'$. Next, we show that either the feasible route in $T_{\mathcal{C}}$ (if exists) can be expanded to an optimal route for $\mathcal{C}$, or there is no feasible route in $\mathcal{C}$ at all. If $T_{\mathcal{C}}$ is not feasible then we are done. Otherwise, let $\mathscr{R}$ be the (unique) route in this tree. For each ring, $\mathscr{R}$ induces some *endpoints* (Fig. 8), one endpoint on each node that is either a) the joint of $T_{\mathcal{C}}$ and the ring, or b) the joint with its adjacent rings. Now we focus on the subproblem induced by this ring and the new waypoint set $\mathscr{W}''$ (to be specified) as follows.

For each endpoints that is visited by $\mathscr{R}$ add a waypoint to $\mathscr{W}''$. Then, using the algorithm described in the proof of Lemma 11, find an optimal route $\mathscr{R}_{ring}$ visiting all the nodes in $\mathscr{W}''$ respecting the order imposed by $\mathscr{R}$. If no such route exists, the instance is not feasible. Otherwise, remove from $\mathscr{R}$ every occurrence of the super node that represents this ring to get a disconnected route. For each missing part, reconnect the endpoints using the segment of $\mathscr{R}_{ring}$ restricted to these endpoints. Repeat this for every ring; denote the resulting route as $\mathscr{R}'$.

Finally, we argue that $\mathscr{R}'$ is optimal. This is the case because its pieces were taken from sets of sub-routes, where each set, covers a disjoint–or more precisely, vertex-disjoint up to endpoints–component of $\mathcal{C}$. Moreover, the set of sub-routes taken from an individual (disjoint) component (i.e. tree or ring) is optimal on that component. Therefore the total length is optimal. □

4 Conclusion

We initiated the study of a natural problem in full-duplex networks: routing through a given set of network functions, the so-called waypoints. We showed

that an optimal routing through a super-constant number of $\mathcal{O}(\log n)$ unordered waypoints can be computed in polynomial time, but that the general optimization problem is NP-hard. Nonetheless, we provided approximation algorithms with small constant competitive ratios for any number of waypoints, via connections to the Steiner Tree and (prize-collecting) Traveling Salesman problems. We also presented hardness results and polynomial-time algorithms for the ordered variant. In particular, we derived an exact polynomial-time algorithm for cactus graphs.

We believe that our work opens several interesting directions for future research. In general, while practically relevant, bidirected networks are not well-understood today, and assume an interesting position between directed and undirected networks. In particular, it would be interesting to understand for which bidirected graph classes the ordered and the unordered waypoint routing problem permits polynomial-time algorithms, and for up to how many waypoints. Another interesting direction for future research concerns the study of randomized algorithms.

Acknowledgements

Klaus-Tycho Foerster is supported by VILLUM FONDEN project ReNet and Mahmoud Parham by AAU's PreLytics project.

References

1. S. Akhoondian Amiri, K.-T. Foerster, R. Jacob, and S. Schmid. Charting the Complexity Landscape of Waypoint Routing. *arXiv preprint arXiv:1705.00055*, 2017.
2. S. Akhoondian Amiri, K.-T. Foerster, and S. Schmid. Walking through waypoints. *arXiv preprint arXiv:1708.09827*, 2017.
3. Aaron Archer, Mohammad Hossein Bateni, Mohammad Taghi Hajiaghayi, and Howard J. Karloff. Improved approximation algorithms for prize-collecting steiner tree and TSP. *SIAM J. Comput.*, 40(2):309–332, 2011.
4. Sanjeev Arora, Michelangelo Grigni, David R. Karger, Philip N. Klein, and Andrzej Woloszyn. A polynomial-time approximation scheme for weighted planar graph TSP. In *Proc. SODA*, 1998.
5. Nikhil Bansal, Kang-Won Lee, Viswanath Nagarajan, and Murtaza Zafer. Minimum congestion mapping in a cloud. In *Proc. PODC*, 2011.
6. Richard Bellman. Dynamic programming treatment of the travelling salesman problem. *J. ACM*, 9(1):61–63, 1962.
7. Andreas Björklund, Thore Husfeld, and Nina Taslaman. Shortest cycle through specified elements. In *Proc. SODA*, 2012.
8. Andreas Björklund and Thore Husfeldt. Shortest two disjoint paths in polynomial time. In *Proc. ICALP*, 2014.
9. Jaroslaw Byrka, Fabrizio Grandoni, Thomas Rothvoß, and Laura Sanità. Steiner tree approximation via iterative randomized rounding. *J. ACM*, 60(1):6:1–6:33, 2013.

10. Pascal Chanas. *Reseaux atm: conception et optimisation.* PhD thesis, University of Grenoble, 1998.
11. Nicos Christofides. Worst-case analysis of a new heuristic for the travelling salesman problem. Technical Report 388, Graduate School of Industrial Administration, Carnegie Mellon University, 1976.
12. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, 3rd Edition.* MIT Press, 2009.
13. Jack Edmonds and Ellis Johnson. Matching: a well-solved class of linear programs. In *Combinatorial Structures and their Applications: Proceedings of the Calgary Symposium*, pages 88–92. Gordon and Breach, New York, 1970.
14. ETSI. Network functions virtualisation – introductory white paper. *White Paper*, oct 2013.
15. Guy Even, Moti Medina, and Boaz Patt-Shamir. Online path computation and function placement in SDNs. In *Proc. SSS*, 2016.
16. Guy Even, Matthias Rost, and Stefan Schmid. An approximation algorithm for path computation and function placement in SDNs. In *Proc. SIROCCO*, 2016.
17. Steven Fortune, John E. Hopcroft, and James Wyllie. The directed subgraph homeomorphism problem. *Theor. Comput. Sci.*, 10:111–121, 1980.
18. M. R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman, 1979.
19. Renaud Hartert, Stefano Vissicchio, Pierre Schaus, Olivier Bonaventure, Clarence Filsfils, Thomas Telkamp, and Pierre Francois. A declarative and expressive approach to control forwarding paths in carrier-grade networks. In *Proc. SIGCOMM*, 2015.
20. Michael Held and Richard M. Karp. The traveling-salesman problem and minimum spanning trees: Part II. *Math. Program.*, 1(1):6–25, 1971.
21. J. Sherry et al. Making middleboxes someone else's problem: Network processing as a cloud service. In *Proc. ACM SIGCOMM*, 2012.
22. A. Jarry and S. Prennes. Disjoint paths in symmetric digraphs. *Discrete Applied Mathematics*, 157(1):90 – 97, 2009.
23. Aubin Jarry. Multiflows in symmetric digraphs. *Discrete Applied Mathematics*, 160(15):2208–2220, 2012.
24. Marek Karpinski, Michael Lampis, and Richard Schmied. New inapproximability bounds for TSP. *J. Comput. Syst. Sci.*, 81(8):1665–1677, 2015.
25. Ken-ichi Kawarabayashi, Yusuke Kobayashi, and Bruce A. Reed. The disjoint paths problem in quadratic time. *J. Comb. Theory, Ser. B*, 102(2):424–435, 2012.
26. Samir Khuller, Stephen G. Mitchell, and Vijay V. Vazirani. Processor efficient parallel algorithms for the two disjoint paths problem and for finding a kuratowski homeomorph. *SIAM J. Comput.*, 21(3):486–506, 1992.
27. Samir Khuller and Baruch Schieber. Efficient parallel algorithms for testing k-connectivity and finding disjoint s-t paths in graphs. *SIAM J. Comput.*, 20(2):352–375, 1991.
28. Philip N. Klein. A subset spanner for planar graphs, : with application to subset TSP. In *Proc. STOC*, 2006.
29. Philip N. Klein and Dániel Marx. A subexponential parameterized algorithm for subset TSP on planar graphs. In *Proc. SODA*, 2014.
30. Tamas Lukovszki and Stefan Schmid. Online admission control and embedding of service chains. In *SIROCCO*, 2015.
31. J. Napper, W. Haeffner, M. Stiemerling, D. R. Lopez, and J. Uttaro. Service Function Chaining Use Cases in Mobile Networks. Internet-draft, IETF, April 2016.

32. Guyslain Naves and András Sebö. Multiflow feasibility: An annotated tableau. In William J. Cook, László Lovász, and Jens Vygen, editors, *Research Trends in Combinatorial Optimization*, pages 261–283. Springer, 2008.
33. P. Sköldström et al. Towards unified programmability of cloud and carrier infrastructure. In *Proc. EWSDN*, 2014.
34. R. Soulé et al. Merlin: A language for provisioning network resources. In *Proc. ACM CoNEXT*, 2014.
35. Neil Robertson and Paul D. Seymour. Graph Minors .XIII. The Disjoint Paths Problem. *J. Comb. Theory, Ser. B*, 63(1):65–110, 1995.
36. Gabriel Robins and Alexander Zelikovsky. Tighter bounds for graph steiner tree approximation. *SIAM J. Discrete Math.*, 19(1):122–134, 2005.
37. Jerome H Saltzer, David P Reed, and David D Clark. End-to-end arguments in system design. *ACM Transactions on Computer Systems (TOCS)*, 2(4):277–288, 1984.
38. András Sebö and Anke van Zuylen. The salesman's improved paths: A 3/2+1/34 approximation. In *Proc. FOCS*, 2016.
39. Rita Vachani, Alexander Shulman, Peter Kubat, and Julie Ward. Multicommodity flows in ring networks. *INFORMS Journal on Computing*, 8(3):235–242, 1996.